REMARKS

This is a full and timely response to the outstanding non-final Office Action mailed June 9, 2009. The Examiner is thanked for the thorough examination of the present application. Upon entry of this response, claims 1-3, 5-14, and 16-22 are pending in the present application. Claims 5 and 16 are amended. Applicants respectfully request consideration of the following remarks contained herein. Reconsideration and allowance of the application and presently pending claims are respectfully requested.

I. Objection to the Abstract

On page 2, the Office Action maintains the objection to the abstract. Applicants submit that the amended abstract meets the requirements set forth under MPEP 608.01(b) and respectfully request that the objection be withdrawn.

II. Response to Claim Rejections Under 35 U.S.C. § 112

Claims 5 and 16 are rejected under 35 U.S.C. § 112, second paragraph. As set forth above, claims 5 and 16 have been amended to address the rejection. Applicants respectfully request that the rejection be withdrawn.

III. Response to Claim Rejections Under 35 U.S.C. § 103

For a proper rejection of the claim under 35 U.S.C. §103, the cited combination of references must disclose, teach, or suggest all elements / features of the claim at issue. See, e.g., *In re Dow Chemical*, 5 U.S.P.Q.2d 1529, 1531 (Fed. Cir. 1988) and *In re Keller*, 208 U.S.P.Q.2d 871, 881 (C.C.P.A. 1981). Claims 1-3, 5-14, and 16-22 are rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over *Ogus et al.* (U.S. patent No. 6,964,046, hereinafter "*Ogus*") in view of *Huynh et al.* (U.S. patent No.

5,386,561, hereinafter "*Huynh*") in further view of *Moore et al.* (U.S. patent No. 5,953,336, hereinafter "*Moore*"). For at least the reasons set forth below, Applicants traverse the rejections set forth.

A. Claims 1-3 and 5-7

Applicants respectfully submit that independent claim 1 patently defines over *Ogus* in view of *Huynh* in further view of *Moore* for at least the reason that the combination fails to disclose, teach, or suggest the features emphasized below in claim 1.

Claim 1 recites:

> 1. A method for scheduling thread execution, comprising:
> maintaining a circular array structure having a plurality of time slots therein, wherein each of the plurality of time slots corresponds to a timeslice during which CPU resources are allocated to a particular thread;
> configuring each time slot in the circular array to include a queue of threads scheduled for execution during that time slot;
> maintaining a pointer index for referencing one time slot in the circular array and whereby advancement through the circular array is provided by advancing the pointer index;
> ***maintaining an array of threads requesting immediate CPU resource allocation based on the queue of threads***;
> suspending a currently executing thread upon expiration of a current timeslice associated with the currently executing thread;
> calculating a next time slot during which the currently executing thread should next resume execution;
> ***appending the suspended currently executing thread to the queue of threads scheduled for execution at the calculated time slot***;
> identifying a next sequential non-empty time slot that includes a queue of threads scheduled for execution during that time slot;
> updating the pointer index to point to the identified next sequential non-empty time slot;
> appending any contents of the indexed time slot to the array of threads requesting immediate CPU resource allocation;
> removing the thread at the top of the array of threads requesting immediate CPU resource allocation; and

activating the thread at the top of the array of threads requesting immediate CPU resource allocation.

(Emphasis added). For the features emphasized above in claim 1, the Office Action relies on the *Huynh* reference and acknowledges that the primary *Ogus* reference fails to disclose these features. Applicants respectfully traverse the rejection.

In alleging that *Huynh* discloses "maintaining an array of threads requesting immediate CPU resource allocation based on the queue of threads," the Office Action cites col. 5, lines 63-68 and col. 6, lines 1-6. *Huynh* is generally directed to processor scheduling for a computer and to regulation of time slice durations allocated processes for execution. More specifically, *Huynh* relates to dynamic regulation of time slice duration as a function of selected operating environment conditions relating to the possible occurrence of memory churning. (*Huynh*, Technical Field section). In col. 5, lines 63-68, *Huynh* teaches of round-robin scheduling, where round-robin is described as a priority preemptive algorithm in which processor time and other resource allocations are sequentially dedicated to each task admitted to a ready to run queue. Each task at a given priority level initially has equal claim to processor time. In col. 6, lines 1-6, *Huynh* further discloses that 128 priority levels are defined and that 32 of these priority levels are associated with four ranked categories of priority. The Office Action alleges that the teachings in these passages correspond to "maintaining an array of threads requesting immediate CPU resource allocation <u>based on the queue of threads</u>" where the queue of threads refers to a <u>queue of threads scheduled for execution</u> during a particular time slot in the circular array, as expressly defined in claim 1. Applicants respectfully disagree. In the passages cited by the Office Action, *Huynh* teaches that processor time and other resource allocations are sequentially

dedicated to each task admitted to a ready to run queue. Each task at a given priority level initially has equal claim to processor time. However, it is not clear how this corresponds to maintaining an *array of threads* requesting immediate CPU resource allocation *based on* the queue of threads. Applicants respectfully submit that *Huynh* fails to disclose or suggest this feature, and that the *Ogus* and *Moore* references fail to address this deficiency.

As an additional basis for patentability, Applicants respectfully submit that the cited references also fail to disclose or suggest "appending the suspended currently executing thread to the queue of threads scheduled for execution at the calculated time slot." For this feature, the Office Action acknowledges that *Ogus* and *Huynh* fail to teach this and relies on *Moore* to teach this feature. *Moore* is directed to scheduling the transmission of cells onto a network or other packet switching network. For the step of appending the suspended currently executing thread, the Office Action apparently alleges that the appending of queues of dynamic actions to a Latent Queue as taught by *Moore* corresponds to this feature. Specifically, in col. 8, lines 34-37, *Moore* teaches the following:

> The Latent Queue is the mechanism by which all dynamic actions which have become current are queued for processing. As queues of dynamic actions are encountered in current ring entries, they are appended to the Latent Queue.

Moore, however, does not appear to disclose or suggest appending the suspended currently executing thread to the queue of threads *scheduled for execution at the calculated time slot*. That is, the suspended thread is appended to a specific queue of threads – those scheduled for execution at the calculated time slot. (Applicants note that claim 1 recites "calculating a next time slot during which the currently executing

thread should next resume execution.") The cited *Moore* reference does not appear to disclose or suggest this feature.

In view of the foregoing, Applicants respectfully submit that independent claim 1 patently defines over *Ogus* in view of *Huynh* in further view of *Moore* for at least the reason that combination fails to disclose, teach, or suggest the highlighted features in claim 1 above. Furthermore, Applicants submit that dependent claims 2, 3 and 5-7 are allowable for at least the reason that these claims depend from an allowable independent claim. *See, e.g., In re Fine*, 837 F. 2d 1071 (Fed. Cir. 1988).

B. Claim 8-10

Applicants respectfully submit that independent claim 8 patently defines over *Ogus* in view of *Huynh* in further view of *Moore* for at least the reason that the combination fails to disclose, teach, or suggest the features emphasized below in claim 8.

Claim 8 recites:

> 8. A method for scheduling thread execution, comprising:
> maintaining a plurality of circular array structures associated with a plurality of discrete thread priorities, each having a plurality of time slots therein, wherein each of the plurality of time slots corresponds to a timeslice during which CPU resources are allocated to a particular thread;
> configuring each time slot in each of the circular arrays to include a queue of threads scheduled for execution during that time slot;
> maintaining at least one pointer index for referencing one time slot in each of the circular arrays, whereby advancement through the circular arrays is provided by advancing the pointer index;
> maintaining an array of threads requesting immediate CPU resource allocation for each of the plurality of circular arrays;
> assigning each thread to be executed a specific priority;
> incrementing the index pointer by one slot;
> removing, for each of the plurality of circular arrays, each

queue of threads for the indexed time slot;

appending each removed thread to the array of threads requesting immediate CPU resource allocation associated with its respective circular array;

determining whether the array of threads requesting immediate CPU resource allocation associated with a first circular array contains any threads;

proceeding to a next circular array if the array of threads requesting immediate CPU resource allocation is empty;

extracting its top thread if the array of threads requesting immediate CPU resource allocation contains any threads;

determining whether a priority of the top thread is greater than a priority of the currently executing thread;

calculating a time for next execution of the top thread if it is determined that the priority of the top thread is not greater than the priority of the currently executing thread;

performing the following steps if it is determined that the priority of the top thread is greater than a priority of the currently executing thread:

suspending the currently executing thread;

activating the top thread; and

calculating the time of next execution for the suspended thread;

determining whether each of the array of threads requesting immediate CPU resource allocation associated with each of the circular arrays has been processed; and

proceeding to the next array of threads requesting immediate CPU resource allocation if it is determined that not all arrays of threads requesting immediate CPU resource allocation have been processed.

(Emphasis added). For the features cited above, the Office Action relies on *Huynh* to teach this feature. Specifically, the Office Action cites col. 6, lines 65-66 for determining whether a priority of the top thread is greater than a priority of the currently executing thread. For the suspending, activating, and calculating features, the Office Action cites col. 6, lines 17-21, FIG. 2, and col. 6, lines 60-66. Applicants respectfully submit that *Huynh* fails to disclose or suggest the features emphasized above. First, in col. 6, lines 65-66, *Huynh* teaches of a "higher priority process" ("[T]he priority preemptive aspect of the scheduling algorithm controls and step 302 is executed,

resulting in the current process being terminated and control being returned to the dispatcher for dispatch of the higher priority process.") For the feature of "suspending the currently executing thread" and "calculating the time of next execution for the suspended thread," the Office Action alleges that these steps correspond to the following in *Huynh*:

> Each task receives a quanta of time and if processing is not completed within that set period the task is removed from the central processing unit 20 and returned to the bottom of the ready to run queue for that priority level.

(Col. 6, lines 16-21). Applicants emphasize, however, the step of suspending the currently executing thread is performed conditionally. Namely, this step is performed **if** it is determined that the priority of the top thread is greater than a priority of the currently executing thread. Applicants respectfully submit that the reference to higher priority processes and the removal of tasks if processing is not completed within its allocated quanta of time does not read on the features emphasized above. In this regard, *Huynh* fails to disclose the conditional execution of the steps above. Furthermore, the remaining references fail to address this deficiency.

Accordingly, Applicants respectfully submit that independent claim 8 patently defines over *Ogus* in view of *Huynh* in further view of *Moore*. Furthermore, Applicants submit that dependent claims 9 and 10 are allowable for at least the reason that these claims depend from an allowable independent claim.

C. Claim 11

Applicants respectfully submit that independent claim 11 patently defines over *Ogus* in view of *Huynh* in further view of *Moore* for at least the reason that the

combination fails to disclose, teach, or suggest the features emphasized below in claim 11.

Claim 11 recites:

> 11. A method for scheduling thread execution, comprising:
> maintaining a circular array structure having a plurality of time slots therein, wherein each of the plurality of time slots corresponds to a timeslice during which CPU resources are allocated to a particular thread;
> configuring each time slot in the circular array to include a queue of threads scheduled for execution during that time slot;
> maintaining a pointer index for referencing one time slot in the circular array and whereby advancement through the circular array is provided by advancing the pointer index;
> ***maintaining an array of threads requesting immediate CPU resource allocation based on the queue of threads***;
> calculating a next time slot during which a currently executing thread should next resume execution;
> ***appending the currently executing thread to the queue of threads scheduled for execution at the calculated time slot***;
> identifying a next sequential non-empty time slot containing a queue of threads scheduled for execution during that time slot;
> updating the pointer index to point to the identified next sequential non-empty time slot;
> appending any contents of the indexed time slot to the array of threads requesting immediate CPU resource allocation;
> removing the thread at the top of the array of threads requesting immediate CPU resource allocation;
> determining whether the thread at the top of the array of threads requesting immediate CPU resource allocation is identical to the currently executing thread;
> maintaining execution of the currently executing thread for the following time slot if it is determined that the thread at the top of the array of threads requesting immediate CPU resource allocation is identical to the currently executing thread;
> suspending a currently executing thread; and
> activating the thread at the top of the array of threads requesting immediate CPU resource allocation if it is determined that the thread at the top of the array of threads requesting immediate CPU resource allocation is not identical to the currently executing thread.

(Emphasis added). In rejecting claim 11, the Office Action cites the same passages in *Huynh* for the features emphasized above. Applicants submit that arguments similar to those set forth above for claim 1 apply. In col. 5, lines 63-68, *Huynh* teaches of round-robin scheduling, where round-robin is described as a priority preemptive algorithm in which processor time and other resource allocations are sequentially dedicated to each task admitted to a ready to run queue. Each task at a given priority level initially has equal claim to processor time. In col. 6, lines 1-6, *Huynh* further discloses that 128 priority levels are defined and that 32 of these priority levels are associated with four ranked categories of priority. The Office Action alleges that the teachings in these passages correspond to "maintaining an array of threads requesting immediate CPU resource allocation based on the queue of threads" where the queue of threads refers to a queue of threads scheduled for execution during a particular time slot in the circular array, as expressly defined in claim 11. Applicants respectfully submit that *Huynh* fails to teach this feature. In the passages cited by the Office Action, *Huynh* teaches that processor time and other resource allocations are sequentially dedicated to each task admitted to a ready to run queue. Each task at a given priority level initially has equal claim to processor time. However, it is not clear how this corresponds to maintaining an array of threads requesting immediate CPU resource allocation based on the queue of threads. Applicants respectfully submit that *Huynh* fails to disclose or suggest this feature, and that the *Ogus* and *Moore* references do not address this deficiency.

As an additional basis for patentability, Applicants respectfully submit that the cited references fail to disclose or suggest "appending the currently executing thread to

the queue of threads scheduled for execution at the calculated time slot." For this feature, the Office Action acknowledges that *Ogus* and *Huynh* fail to teach this feature and relies on the *Moore* reference. For the step of appending the currently executing thread, the Office Action apparently alleges that the appending of queues of dynamic actions to a Latent Queue as taught by *Moore* corresponds to this feature. *Moore*, however, does not appear to disclose or suggest appending the currently executing thread to the queue of threads *scheduled for execution at the calculated time slot*. That is, the suspended thread is appended to a specific queue of threads – those scheduled for execution at the calculated time slot. (Applicants note that claim 11 recites "calculating a next time slot during which a currently executing thread should next resume execution.") *Moore* does not appear to disclose or suggest this feature.

In view of the foregoing, Applicants respectfully submit that independent claim 11 patently defines over *Ogus* in view of *Huynh* in further view of *Moore* for at least the reason that combination fails to disclose, teach, or suggest the highlighted features in claim 11 above.

D. Claims 12-14, 16-18, 19-22

On page 21, the Office Action relies on the same rationale used in rejecting claims 1-3, 5-7, and 8-11 to reject claims 12-14, 16-18, and 19-22. As independent claims 12, 19, and 22 embody similar defining features as independent claims 1, 8, and 11, Applicants respectfully submit that these claims define over the cited art for similar reasons. Insofar as claims 13, 14, 16-18, 20, and 21 depend from claims 12 and 19, these claims are also allowable. *See, e.g., In re Fine*, 837 F. 2d 1071 (Fed. Cir. 1988).

CONCLUSION

Applicants respectfully submit that all pending claims are in condition for allowance. Favorable reconsideration and allowance of the present application and all pending claims are hereby courteously requested. If, in the opinion of the Examiner, a telephone conference would expedite the examination of this matter, the Examiner is invited to call the undersigned attorney at (770) 933-9500.

No fee is believed to be due in connection with this amendment and response to Office Action. If, however, any fee is believed to be due, you are hereby authorized to charge any such fee to deposit account No. 50-0835.

Respectfully submitted,

/ Jeffrey Hsu /

Jeffrey C. Hsu
Reg. No. 63,063

THOMAS, KAYDEN, HORSTEMEYER & RISLEY, L.L.P.
600 Galleria Parkway SE
Suite 1500
Atlanta, Georgia 30339
(770) 933-9500